UNIVERSITY OF TWENTE.

On the computation of the Walsh-Hadamard Transform using Binomial Trees

Luca Mariot

l.mariot@utwente.nl

Joint work in progress with Alberto Leporati, Stjepan Picek and Claude Carlet

SymCrypt24@CIFRIS'24 – Rome, September 27, 2024

Representations of Boolean Functions

- **Truth table**: given $f : \mathbb{F}_2^n \to \mathbb{F}_2$, vector Ω_f specifying $f(x)$ for all $x \in \mathbb{F}_2^n$
- **Fourier transform**: correlation with *linear* functions $a \cdot x = ax_1 \oplus \cdots \oplus a_n x_n$, $a \in \mathbb{F}_2^n$:

$$\hat{f}(a) = \sum_{x \in \mathbb{F}_2^n} f(x) \cdot (-1)^{a \cdot x}$$

- **Walsh Transform**: Fourier transform of the sign function of f

$$W_f(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus a \cdot x}$$

(x_0, x_1, x_2)	000	100	010	110	001	101	011	111
Ω_f	0	1	1	0	1	0	0	1
$\hat{f}(a)$	0	0	0	0	0	0	0	4
$W_f(a)$	0	0	0	0	0	0	0	8

Example: $n = 3$, $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$

Correlation Immunity: Walsh Characterization

- f is t-**Correlation Immune** (t-CI) iff $W_f(a) = 0$ for all a s.t. $1 \leq w_H(a) \leq t$, where $w_H(a)$ is the Hamming weight of a [X88]

Example: $t = 2$

(x_0, x_1, x_2)	000	**100**	**010**	**110**	**001**	**101**	**011**	111
Ω_f	0	1	1	0	1	0	0	1
$\hat{f}(a)$	0	0	0	0	0	0	0	4
$W_f(a)$	0	0	0	0	0	0	0	8

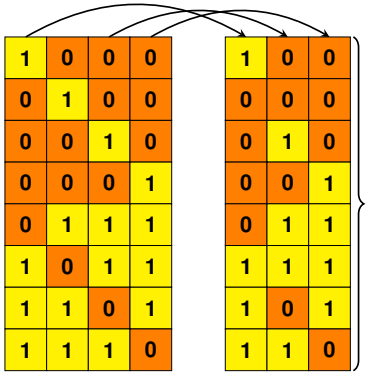
$\Downarrow$

f is 2-CI

- Relevance in side-channel: t-CI functions $\Rightarrow$ Boolean masking resistant to SCA attacks of order t [C21]

Orthogonal Arrays (OA)

- (N,k,s,t) **Orthogonal Array**: $N \times k$ matrix A such that each t-uple occurs $\lambda = N/s^t$ times in each $N \times t$ submatrix.



Example: OA $(8,4,2,3)$

$\Rightarrow$ Each 3-bit vector $(x_1,x_2,x_3) \in \{0,1\}^3$ appears once in the submatrix with columns 1, 3, 4

- **Motivation** for this work: more efficient fitness function to construct OA with *Evolutionary Algorithms* [M18, M20, M21]

Correlation Immunity: OA Characterization

- **Support** of f: $Supp(f) = \{x \in \mathbb{F}_2^n : f(x) \neq 0\}$

Truth table

x_0	x_1	x_2	$f(x)$
0	0	0	0
0	0	1	**1**
0	1	0	**1**
0	1	1	0
1	0	0	**1**
1	0	1	0
1	1	0	0
1	1	1	**1**

Support

x_0	x_1	x_2
0	0	1
0	1	0
1	0	0
1	1	1

$\Downarrow$

$OA(4,3,2,2)$

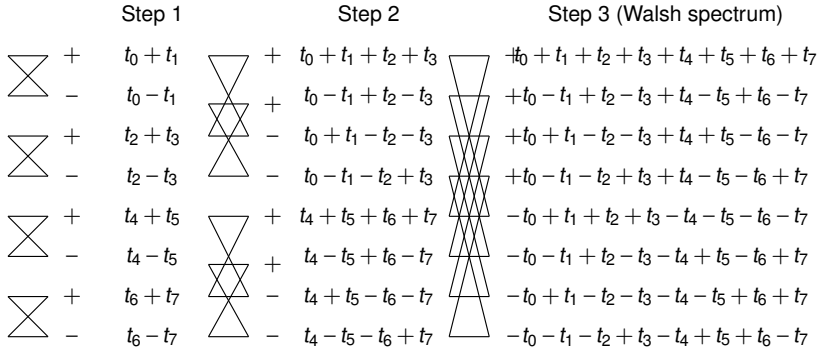
Theorem ([C92])

$f : \{0,1\}^n \to \{0,1\}$ is t-order CI $\Leftrightarrow$ Support of f is an $OA(N,n,2,t)$, with $N = |Supp(f)|$

Walsh Transform Computation

- **Naive algorithm**: time complexity $O(2^{2n})$
- **Fast Walsh Transform** (FWT): time complexity $O(n2^n)$ [C21]

x_0	x_1	x_2	$\hat{f}$
0	0	0	t_0
0	0	1	t_1
0	1	0	t_2
0	1	1	t_3
1	0	0	t_4
1	0	1	t_5
1	1	0	t_6
1	1	1	t_7



Problem Statement

Question: what if we want to check correlation immunity of a function f with a large number of variables, say $n = 80$?

- We don't need the whole Walsh spectrum: just the coefficients up to weight t...
- ...but if n is large, we can't even store the truth table of f!

$$n = 80 \Rightarrow 2^{80} \text{ bits } \approx 1.51 \cdot 10^{11} \text{ terabytes}$$

- Both the naive and fast Walsh transforms need the whole truth table

Fourier Transform from the Support

Remark

Given $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and $a \in \mathbb{F}_2^n$, we have:

$$\hat{f}(a) = \sum_{x \in \mathbb{F}_2^n} f(x) \cdot (-1)^{a \cdot x} = \sum_{x \in supp(f)} (-1)^{a \cdot x}$$

- **Consequence:** if n is large, but $|supp(f)|$ is small enough, we can still check t-CI with the naive procedure restricted to $supp(f)$
- **Question:** can we do better than the naive transform?

Step 1: Column-wise Scalar Product

- In $a \cdot x$, $supp(a)$ selects the coordinates of x:

$$a \cdot x = \bigoplus_{i \in supp(a)} x_i$$

- So we can rewrite the Fourier coefficients as:

$$\hat{f}(a) = \sum_{x \in supp(f)} (-1)^{\bigoplus_{i \in supp(a)} x_i}$$

Example: $a = (1,0,1)$

$\mathbf{x_1}$	x_2	$\mathbf{x_3}$	$a \cdot x$	$(-1)^{a \cdot x}$
0	0	**1**	1	**-1**
0	1	**0**	0	**+1**
1	0	**0**	1	**-1**
1	1	**1**	0	**+1**

$\hat{f}(a) = 0$

- **Thus:** $\hat{f}$ is obtained by XORing the columns in $supp(f)$ indexed by $supp(a)$
- $\Rightarrow$ amenable to parallelization (SIMD operations)

Step 2: Sorting Coefficients by Weight

- **Idea:** Compute Fourier coefficients in increasing order of Hamming weight...
- ... and for t-CI, compute them up to weight t
- But is this efficient?

Example: $a = (1,1,1)$

$\mathbf{x_0}$	$\mathbf{x_1}$	$\mathbf{x_2}$	$a \cdot x$	$(-1)^{a \cdot x}$
0	**0**	**1**	1	**-1**
0	**1**	**0**	1	**-1**
1	**0**	**0**	1	**-1**
1	**1**	**1**	1	**-1**

- **Problem:** We already XORed the columns x_0 and x_2 previously for $\hat{f}(1,0,1)$!

Improvement: memoize intermediate XORs with *binomial trees*

Step 3: Enter Binomial Trees

Definition ([K08])

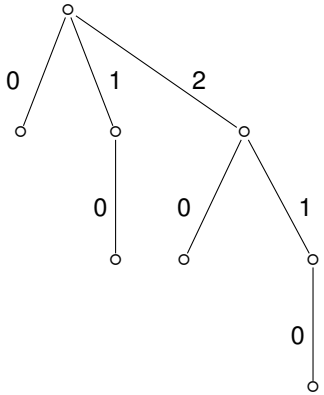
Given $n \in \mathbb{N}$, the *binomial tree* T_n is recursively defined as:

$$T_n = \begin{cases} \circ\ , & \text{if } n = 0\ , \\ \circ \text{ with edges } 0, 1, \ldots, n-1 \text{ to } T_0, T_1, \ldots, T_{n-1}\ , & \text{if } n > 0\ . \end{cases} \tag{1}$$

- At each depth level k, there are exactly $\binom{n}{k}$ nodes
- A path from a node at level k back to the root gives a $(n-k, k)$-combination...
- ... or equivalently a coefficient $a \in \mathbb{F}_2^n$ with $w_h(a) = k$

XOR Memoization with Binomial Trees

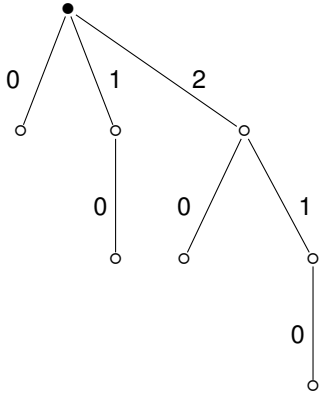
Idea: XOR the column indexed by the edge with the vector inherited from the parent



x_0	x_1	x_2	$a \cdot x$	$(-1)^{a \cdot x}$
0	0	1		
0	1	0		
1	0	0		
1	1	1		
				$\hat{f}(a) =$

XOR Memoization with Binomial Trees

Idea: XOR the column indexed by the edge with the vector inherited from the parent



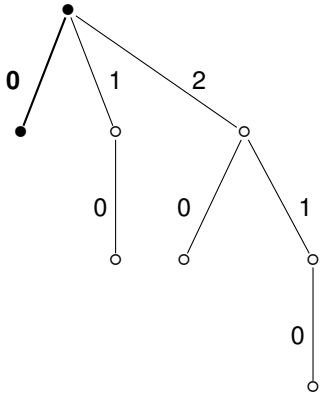
Weight: 0, $a = (0,0,0)$

x_0	x_1	x_2	$a \cdot x$	$(-1)^{a \cdot x}$
0	0	1	0	**+1**
0	1	0	0	**+1**
1	0	0	0	**+1**
1	1	1	0	**+1**

$\hat{f}(a) = \mathbf{4}$

XOR Memoization with Binomial Trees

Idea: XOR the column indexed by the edge with the vector inherited from the parent



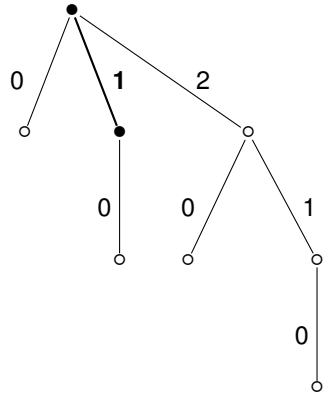
Weight: 1, $a = (1,0,0)$

$\mathbf{x_0}$	x_1	x_2	$a \cdot x$	$(-1)^{a \cdot x}$
0	0	1	0	**+1**
0	1	0	0	**+1**
1	0	0	1	**-1**
1	1	1	1	**-1**

$\hat{f}(a) = \mathbf{0}$

XOR Memoization with Binomial Trees

Idea: XOR the column indexed by the edge with the vector inherited from the parent



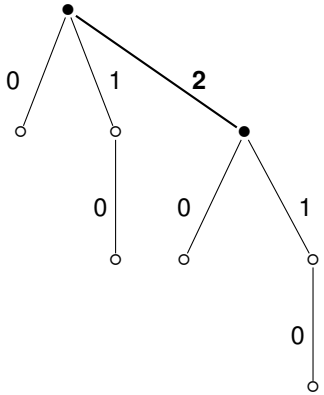
Weight: 1, $a = (0,1,0)$

x_0	$\mathbf{x_1}$	x_2	$a \cdot x$	$(-1)^{a \cdot x}$
0	**0**	1	0	**+1**
0	**1**	0	1	**-1**
1	**0**	0	0	**+1**
1	**1**	1	1	**-1**

$\hat{f}(a) = \mathbf{0}$

XOR Memoization with Binomial Trees

Idea: XOR the column indexed by the edge with the vector inherited from the parent



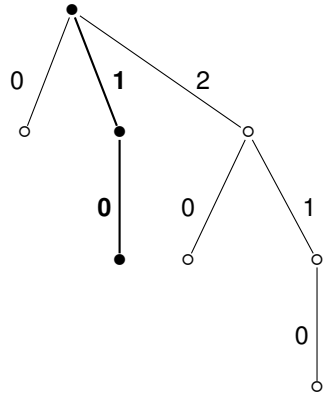
Weight: 1, $a = (0,0,1)$

x_0	x_1	$\mathbf{x_2}$	$a \cdot x$	$(-1)^{a \cdot x}$
0	0	**1**	1	**-1**
0	1	**0**	0	**+1**
1	0	**0**	0	**+1**
1	1	**1**	1	**-1**

$\hat{f}(a) = \mathbf{0}$

XOR Memoization with Binomial Trees

Idea: XOR the column indexed by the edge with the vector inherited from the parent



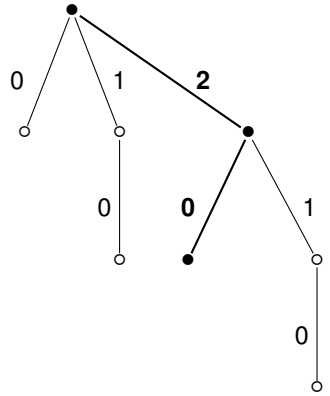
Weight: 2, $a = (1,1,0)$

$\mathbf{x_0}$	$\mathbf{x_1}$	x_2	$a \cdot x$	$(-1)^{a \cdot x}$
0	**0**	1	0	**+1**
0	**1**	0	1	**-1**
1	**0**	0	1	**-1**
1	**1**	1	0	**+1**

$\hat{f}(a) = \mathbf{0}$

XOR Memoization with Binomial Trees

Idea: XOR the column indexed by the edge with the vector inherited from the parent



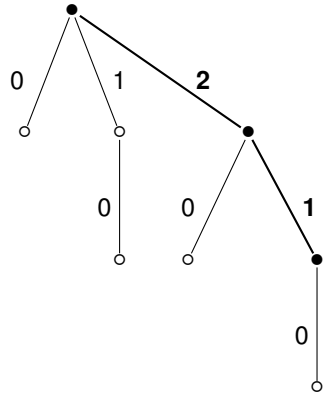
Weight: 2, $a = (1,0,1)$

$\mathbf{x_0}$	x_1	$\mathbf{x_2}$	$a \cdot x$	$(-1)^{a \cdot x}$
0	0	**1**	1	**-1**
0	1	**0**	0	**+1**
1	0	**0**	1	**-1**
1	1	**1**	0	**+1**

$\hat{f}(a) = \mathbf{0}$

XOR Memoization with Binomial Trees

Idea: XOR the column indexed by the edge with the vector inherited from the parent



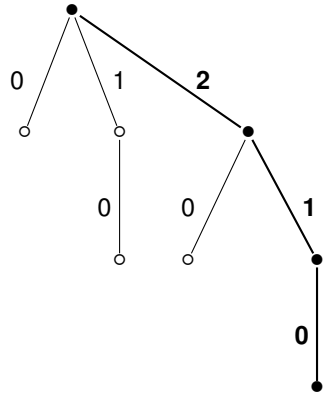
Weight: 2, $a = (0,1,1)$

x_0	$\mathbf{x_1}$	$\mathbf{x_2}$	$a \cdot x$	$(-1)^{a \cdot x}$
0	**0**	**1**	1	**-1**
0	**1**	**0**	1	**-1**
1	**0**	**0**	0	**+1**
1	**1**	**1**	0	**+1**

$\hat{f}(a) = \mathbf{0}$

XOR Memoization with Binomial Trees

Idea: XOR the column indexed by the edge with the vector inherited from the parent

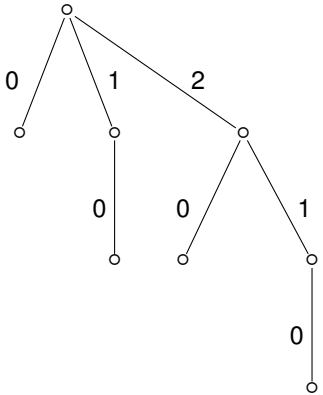


Weight: 3, $a = (1,1,1)$

$\mathbf{x_0}$	$\mathbf{x_1}$	$\mathbf{x_2}$	$a \cdot x$	$(-1)^{a \cdot x}$
0	**0**	**1**	1	**-1**
0	**1**	**0**	1	**-1**
1	**0**	**0**	1	**-1**
1	**1**	**1**	1	**-1**

$\hat{f}(a) = -\mathbf{4}$

Time Complexity

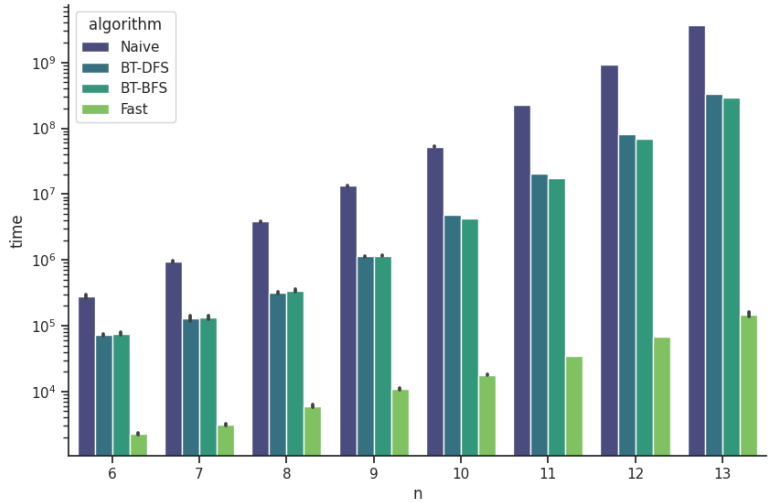


- How to visit T_n: **Breadth-First Search** (BFS) or **Depth-First Search** (DFS)
- Number of nodes visited up to level t: $\sum_{i=0}^{t} \binom{n}{i}$
- If $t = n$ (full spectrum): all 2^n nodes are visited
- Thus: asymptotic time complexity is *still* $O(2^{2n})$ as in the naive procedure

But, what about practical runtimes?

Runtime Experiments

Runtimes measured in *ns* over a sample of $N = 1000$ random functions for $6 \leq n \leq 13$:



Summary

Conclusions:

- Our algorithm has the same asymptotic complexity of the naive transform...
- ... But it is faster in practice!
- Alternative to check t-CI of large functions, where the FWT cannot be used

Future Directions:

- Parallel visit of the binomial tree
- Use to search for OA, codes and t-CI functions with EA [M18, C22, M22]
- Comparing to similar algorithms for partial Walsh transform [G09]
- Application in other use cases (FLIP-like ciphers?) [M16]

References

[C92] P. Camion, C. Carlet, P. Charpin, N. Sendrier: On Correlation-Immune Functions. Proceedings of CRYPTO 1991, pp. 86-100 (1992)

[C22] C. Carlet, L. Mariot, L. Manzoni, S. Picek: Evolutionary Strategies for the Design of Binary Linear Codes. In: Proceedings of EvoCOP 2023. LNCS vol. 13987, pp. 114-129. Springer (2022)

[C21] C. Carlet: Boolean functions for cryptography and coding theory. Cambridge University Press (2021)

[G09] K.C. Gupta, P. Sarkar: Computing Partial Walsh Transform From the Algebraic Normal Form of a Boolean Function. IEEE Trans. Inf. Theory 55(3): 1354-1359 (2009)

[H12] A.S. Hedayat, N.J.A. Sloane, J. Stufken: Orthogonal arrays: theory and applications. Springer Science & Business Media (2012)

[K08] D. E. Knuth: The art of computer programming: Introduction to combinatorial algorithms and Boolean functions. Vol. 4, Addison-Wesley (2008)

[M20] L. Manzoni, L. Mariot, E. Tuba: Balanced crossover operators in Genetic Algorithms. Swarm Evol. Comput. 54: 100646 (2020)

[M22] L. Mariot, L. Manzoni: Building Correlation Immune Functions from Sets of Mutually Orthogonal Cellular Automata. In: Proceedings of AUTOMATA 2022. LNCS vol. pp. 153–164. Springer (2022)

[M21] L. Mariot: Deriving Smaller Orthogonal Arrays from Bigger Ones with Genetic Algorithm. CoRR abs/2111.13047 (2021)

[M18] L. Mariot, S. Picek, D. Jakobovic, A. Leporati: Evolutionary Search of Binary Orthogonal Arrays. In: Proceedings of PPSN 2018 (I). LNCS vol. 11101, pp. 121–133. Springer (2018)

[M16] P. Méaux, A. Journault, F.-X. Standaert, C. Carlet: Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In: Proceedings of EUROCRYPT (1) 2016. LNCS vol. 9665, pp. 311-343. Springer (2016)

[X88] G.-Z. Xiao, J. L. Massey: A spectral characterization of correlation-immune combining functions. IEEE Trans. Inf. Theory 34(3): 569-571 (1988)